SEGURANÇA E GERENCIAMENTO DE USUÁRIOS

LABORATÓRIO DE BANCO DE DADOS

VANESSA BORGES

Segurança em banco de dados

- Uma das maiores preocupações em computação tem sido **segurança da informação**

- Nos dias atuais, com o uso da Internet os sistemas tornam-se onipresentes, entretanto também **vulneráveis** a ataques maliciosos

- Portanto, os SGBDs trazem **uma camada de segurança** que visa compor o arsenal de segurança da informação numa corporação

Mecanismos de segurança

- Um SGBD possui um subsistema de segurança e autorização do banco de dados responsável por **garantir a segurança de partes de um banco de dados** contra acesso não autorizado
- **Formas de acesso não autorizado**:
  - leitura não autorizada
  - modificação não autorizada
  - remoção de dados não autorizada
- O **DBA** (*Database Administrator*, ou super user) tem plenos poderes para dar e revogar privilégios a usuários
  - Criação de contas
  - Concessão/Revogação de privilégios
  - Definição do nível de segurança

Principais ameaças ao banco de dados

- **Perda de Integridade**
  - Proteção contra modificações impróprias
  - A integridade é perdida se mudanças não autorizadas forem feitas nos dados por atos intencionais ou acidentais
- **Perda de Disponibilidade**
  - Tornar os objetos disponíveis a um usuário humano ou a um programa ao qual eles têm um direito legítimo
- **Confidencialidade**
  - Proteção dos dados contra exposição não autorizada

SEGURANÇA E GERENCIAMENTO DE USUÁRIOS

LABORATÓRIO DE BANCO DE DADOS

VANESSA BORGES

Criação de usuário

- O criador de um objeto é o **dono do objeto** e assim tem todos os privilégios sobre o objeto, podendo autorizar a outros usuários alguns (ou todos) destes privilégios

- Definição de um usuário

  CREATE USER <usuario> WITH PASSWORD <senha>;

  - Alguns SGBDs permitem que o usuário use o **mesmo login e senha do SO**
    - **Simplifica** a autenticação

- Quando um usuário é criado, ele **tem associado a ele um conjunto de objetos** dos quais ele é dono e sobre os quais pode definir o controle de acesso

- Remover usuário

  DROP USER <usuario>;

https://www.postgresql.org/docs/current/sql-createrole.html

Criação de usuário

CREATE USER <nome usuário> [[WITH] <opções ...>];

- Onde **opções** :
  - SUPERUSER | NOSUPERUSER
    - Determinam se o novo papel é um "super-usuário", o qual pode passar por cima de todas as restrições de acesso dos bancos de dados. Se nenhuma dessas duas cláusulas for especificada, o padrão é NOSUPERUSER.
  - CREATEDB | NOCREATEDB
    - Definem a permissão para o usuário criar bancos de dados. Se nenhuma destas duas cláusulas for especificada, o padrão é NOCREATEDB

Criação de usuário

CREATE USER <nome usuário> [[WITH] <opções ...>];

- Onde **opções** :
  - INHERIT | NOINHERIT:
    - Determinam se o papel "herda" os privilégios dos papéis dos quais é membro. Um papel com o atributo INHERIT pode utilizar, automaticamente, todos os privilégios de banco de dados que foram concedidos a todos os papéis dos quais é um membro direto ou indireto.
  - LOGIN | NOLOGIN
    - Determinam se o papel pode estabelecer uma conexão (login), ou seja, se o papel pode ser fornecido como nome de autorização inicial da sessão durante a conexão do cliente.
    - Um papel com o atributo LOGIN pode ser considerado como sendo um usuário. Se nenhuma dessas duas cláusulas for especificada, o padrão é NOLOGIN.

Criação de usuário

```
CREATE USER <nome usuário> [ [ WITH ] <opções ...>];
```

- Onde **opções** :
  - CREATEROLE | NOCREATEROLE
    - Determinam se o papel terá permissão para criar novos papéis (ou seja, executar o comando CREATE ROLE)
  - ...

Controle de acesso baseado em papéis

- Papéis (Roles)
  - É um identificador ao qual atribui-se um conjunto de privilégios; um papel pode ser associado a diferentes usuários
  - Pode-se inclusive ao criar um papel usar outros papéis já cadastrados

  - Sintaxe de criação de papel (ROLE)

    CREATE ROLE <nome do papel> [[WITH] PASSWORD '<senha>'];

  - Sintaxe de remoção de papel (ROLE):

    [DROP| DESTROY] ROLE <nome do papel>;

Usuários e Papéis no PostgreSQL

- No PostgreSQL as permissões são gerenciadas usando o conceito de papeis (*roles*).
- O conceito de papéis agrupa os conceitos de usuário
  - Um papel pode atuar como usuário, grupo ou ambos

- Antes da versão 8.1 havia distinção entre os conceitos, que foi extinta nas versões posteriores.
- Os papéis podem ser listados usando o comando:
  - **\du**

```
SELECT * FROM pg_roles;
```

Controle de acesso baseado em papéis

- Papéis (Roles)
  - **É um identificador ao qual atribui-se um conjunto de privilégios**; um papel pode ser associado a **diferentes usuários**
  - Pode-se inclusive ao criar um papel usar outros papéis já cadastrados

  - Exemplo de criação de papel (ROLE):

    - Cria o papel desenvolvedor

      ```
      CREATE ROLE desenvolvedores;
      ```

    - Apaga o papel desenvolvedor

      ```
      DROP ROLE desenvolvedores;
      ```

Adicionar usuários a um papel

- Atribui novas permissões a um papel
  - Atribui permissão de criação de banco de dados ao papel desenvolvedores

    ```
    GRANT CREATEDB TO desenvolvedores;
    ```

  - Atribui permissão para criar funções ao papel desenvolvedores

    ```
    GRANT CREATE PROCEDURE TO desenvolvedores;
    ```

  - Atribui permissões de SELECT E UPDATE na tabela ”tabela01” para o papel desenvolvedores

    ```
    GRANT SELECT, UPDATE ON tabela01 TO desenvolvedores;
    ```

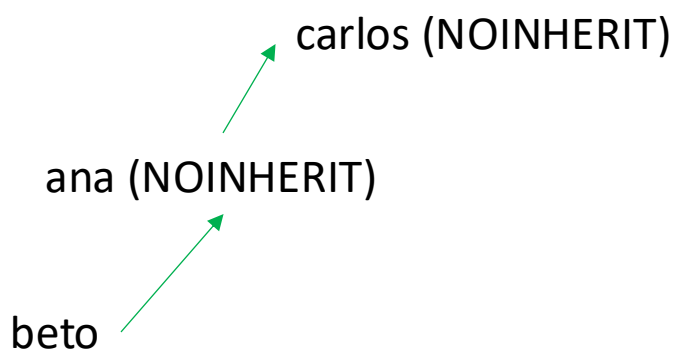
Adicionar usuários a um papel

- Atribui papéis a usuários
  - Membros de papeis que possuem o atributo **INHERIT automaticamente herdam as permissões dos papeis do quais são membros**.
  - Por exemplo, supondo que executamos:

```
CREATE ROLE beto LOGIN INHERIT;
CREATE ROLE ana NOINHERIT;
CREATE ROLE carlos NOINHERIT;
GRANT ana TO beto;
GRANT carlos TO ana;
```

Depois de se conectar como papel beto, a sessão do banco de dados fara uso dos privilégios concedidos a beto mais as do papel ana.

ana foi criada com o atributo NOINHERIT que faz com que beto não herde as permissões herdadas por ana



Usuários e Papéis no PostgreSQL

- Para usar os privilégios concedidos a um grupo pode-se utilizar o comando SET ROLE
- Ele **atribui temporariamente para a sessão as permissões** do grupo ao usuário conectado, ao invés do papel original do usuário
- Qualquer objeto criado é considerado propriedade do grupo e não do usuário

- Sintaxe: SET ROLE <nome do papel>;

- Para voltar ao estado original, qualquer uma das declarações a seguir pode ser executada:

```
SET ROLE <nome do papel anterior>;
SET ROLE NONE;
RESET ROLE;
```

Usuários e Papéis no PostgreSQL

- Se o usuário **beto estiver conectado e executar a declaração**

```
SET ROLE carlos;
```

- O usuário beto passa a ter os privilégios concedidos para o papel/grupo carlos, mesmo não herdando diretamente estes privilégios.
- Os privilégios de beto e ana não são concedidos.

- Os atributos **LOGIN, SUPERUSER, CREATEDB e CREATEROLE** podem ser vistos como atributos especiais, que **não são herdados automaticamente**.
  - O comando SET ROLE deve ser usado para que estes atributos tenham efeito

- Para os exercícios a seguir, escreva as declarações SQL que executem as operações desejadas no PostgreSQL:

1. Crie um usuário ”usuarioalunoX” com a senha 1234; Esse usuário não deverá ter as permissões de superusuário, mas poderá criar database e schema e papéis.

2. Fação um login no psql com esse usuário;

Privilégios: controle de acesso

- Quando um objeto é criado, é atribuído um proprietário que normalmente é o papel que executou a declaração de criação.
- O estado inicial do objeto permite somente o proprietário e o ”super-usuário” acessar o objeto.
- Para permitir que outros papeis acessem o objeto, o comando **GRANT** deve ser usado juntamente com os seguintes tipos de privilégios:
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - TRUNCATE
  - TRIGGER
  - CREATE

Privilégios: controle de acesso

- O comando **GRANT** é utilizado para conferir autorização
  - A forma básica do comando é:

```
GRANT <lista de privilégios> | ALL PRIVILEGES
ON <nome da relação ou view>
TO <lista de usuários | lista de papéis | PUBLIC>
[WITH GRANT OPTION];
```

- A DDL possui o comando **GRANT** para conceder e **REVOKE** para revogar privilégios
- O privilégio **ALL PRIVILEGES** pode ser utilizado como um atalho para todos os privilégios permitidos
- O nome de usuário **PUBLIC** se refere a todos os usuários atuais e futuros do sistema
  - Privilégios concedidos a PUBLIC são implicitamente concedidos a todos os usuários atuais e futuros
- **WITH GRANT OPTION** permite a propagação dos privilégios

Privilégios: controle de acesso

- **Privilégio SELECT:** é exigido para ler tuplas da relação

```
-- Concede aos usuários Ana e João o privilégio de SELECT na relação departamento
GRANT SELECT ON departamento TO Ana, João;
```

- **Privilégio UPDATE:** é exigido para a atualização de tuplas da relação considerando todos os atributos da relação ou apenas alguns
  - Pode especificar uma lista de atributos

```
-- Concede aos usuários Ana e João o privilégio de UPDATE no atributo orçamento da relação departamento
GRANT UPDATE (orçamento) ON departamento TO Ana, João;
```

Privilégios: controle de acesso

- **Privilégio INSERT:** permite que o usuário insira tuplas em uma relação
  - **Pode especificar uma lista de atributos**
  - Quaisquer inserções na relação precisa especificar apenas esses atributos e o sistema fornece valores padrão a cada um dos atributos restantes (se um padrão estiver definido para o atributo) ou os define como nulos

```
GRANT INSERT(id, orçamento, dnome) ON departamento TO Ana, João;
```

- **Privilégio DELETE:** permite que o usuário exclua tuplas de uma relação

```
GRANT DELETE ON departamento TO Ana;
```

Privilégios: controle de acesso
Propagação de privilégios

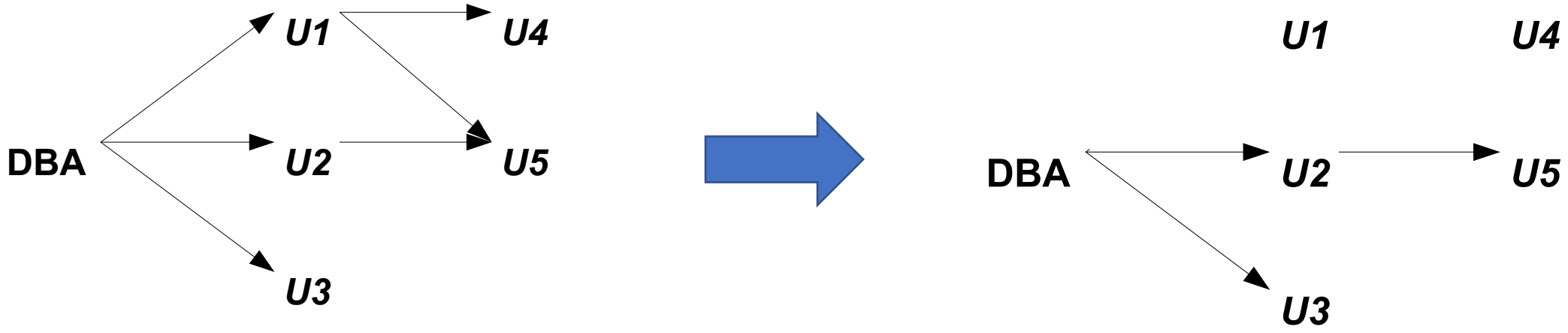
- O DBA fornece/revoga as autorizações de leitura, inserção, atualização e remoção aos usuários nas diversas tabelas/visões, e estes podem repassá-los caso recebam autorização para tal – **WITH GRANT OPTION**
- Exemplo:

  DBA: **GRANT SELECT ON** tabela_de_produtos **TO** U2 **WITH GRANT OPTION**;

  U2: **GRANT SELECT ON** tabela_de_produtos **TO** U3;

Privilégios: controle de acesso
Propagação de privilégios

- Suponha que o administrador do banco de dados decida **revogar a autorização** do usuário *U1*

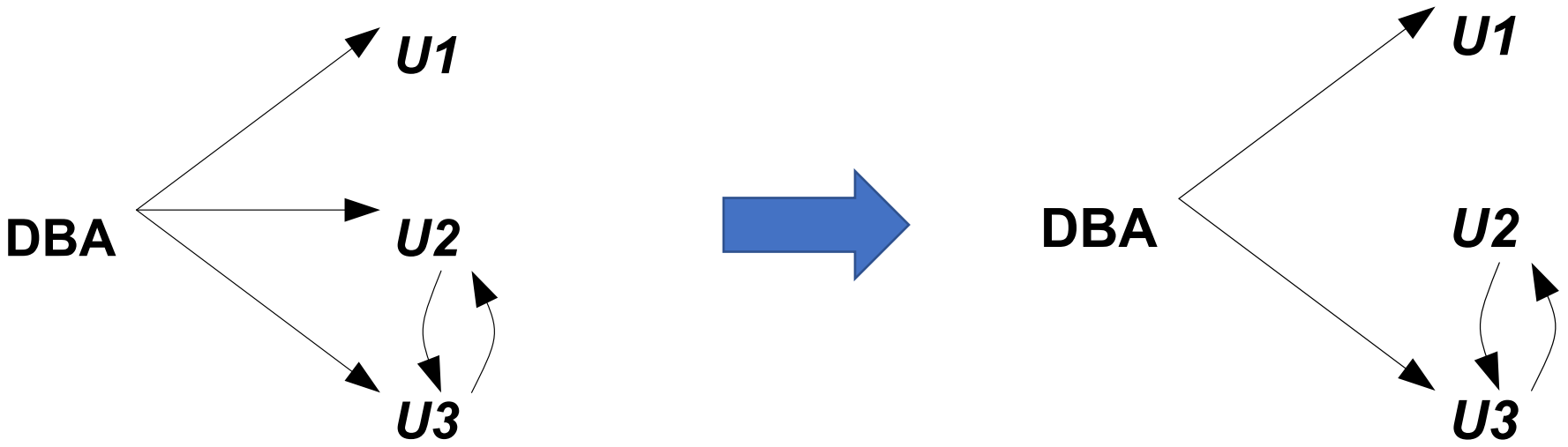


- Uma vez que o usuário *U4* tem a autorização concedida pelo usuário *U1*, a sua autorização também será revogada.

- No entanto, *U5* mantém sua autorização por ela ter sido concedida também por *U2*.

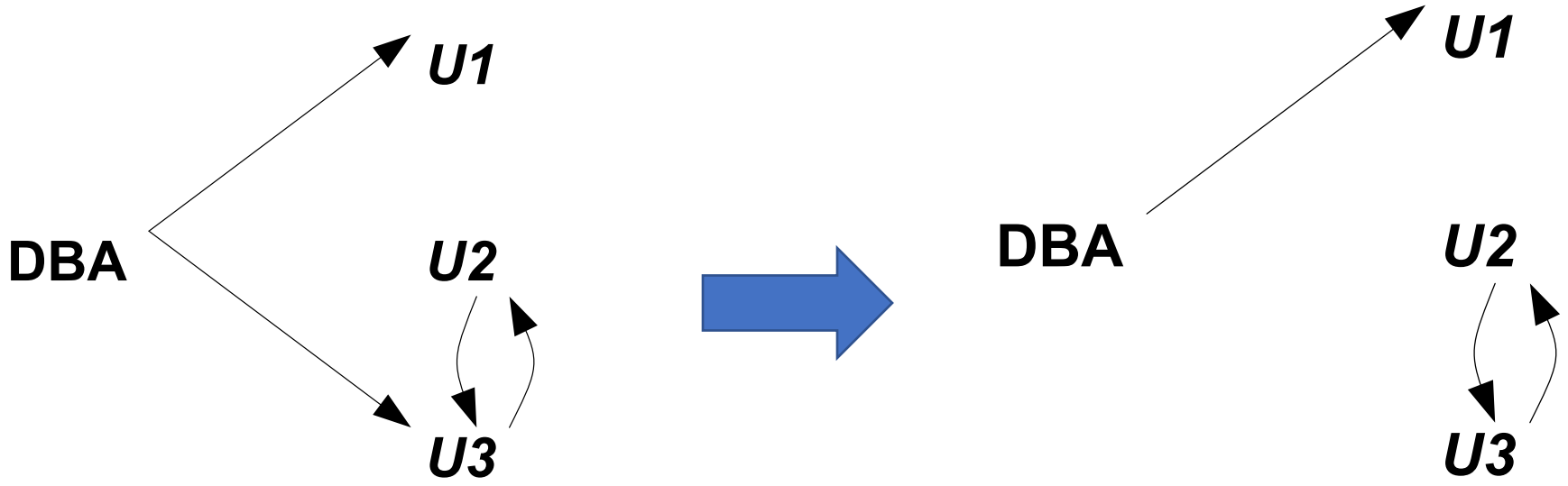
Privilégios: controle de acesso
Propagação de privilégios

- Um par de usuário desonestos pode **tentar burlar** as regras anteriores de revogação de autorização **concedendo autorização de um para outro**
- Se o administrador do banco de dados revogar a autorização de *U2*, este **manteria sua autorização por meio de *U3***



Privilégios: controle de acesso
Propagação de privilégios

- Se a autorização for revogada subsequentemente de U3, ele reteria sua autorização por meio de U2.



Privilégios: controle de acesso
Propagação de privilégios

- Para evitar problemas como esse, os SGBDs são projetados de maneira que **todas as arestas em um grafo de autorização sejam parte de algum caminho originado no administrador do banco de dados**